AMENDMENTS IN THE SPECIFICATION

Please amend the paragraph starting on page 8, line 6 as follows:

To achieve the above objects of the present invention, there is provided a method for generating one primary scrambling code assigned to a base station and multiple secondary scrambling codes with two m-sequence generators each having plurality of concatenated shift registers, the method including the steps of: generating a first m-sequence_by first m-sequence generator having a given generation polynomial_and a second m-sequence by second m-sequence generator having a given generation polynomial different from the first m-sequence generation polynomial; adding the output of the first m-sequence generator and the output of the second m-sequence generator to generate first primary scrambling code for generating primary scrambling code; receiving all values of a first m-sequence registers; multiplying the first m-sequence register values with a mask value which is determining secondary scrambling code and summing the multiplied values at every clock signal; and generating ~~i'th~~ i-th secondary scrambling code by adding the summed value and second m-sequence generator's output.

Please amend the paragraph starting on page 14, line 16 as follows:

The adder 740 adds the ~~0'th~~ 0-th register values_(i.e., the last bits) of the first and second shift register memories 700 and 705 to generate a scrambling code, which becomes the primary scrambling code. The adders 742 to 744 add one bit generated from each of the masking sections 710 to 712 connected to the first shift register memory 700 to one bit generated from the masking sections 714 to 716 corresponding to the masking sections 710 to 712, respectively. In other words, the output from the first masking section 710 from the first group is added with the output from the first masking section 714 from the second group and so on, until the output from the N-th ~~Nth~~ masking section 712 from the first group is added with the output from the N-th ~~Nth~~ masking section 716 from the second group. Thus, each of the masking sections 710 – 712 in the first group has a corresponding masking section in the masking sections 714 – 716 of the second group. The outputs from the corresponding masking sections are added together in the adders 742 – 744, respectively. That is, the individual masking sections have a conjugate on a one-to-one basis with respect to the first and second shift register memories 700 and 705. For example, the first masking section 710 of the first shift register memory 700 corresponds to the first masking section 714 of the second shift register memory 705, the ~~N'th~~ N-th masking section 712 corresponding to the ~~N'th~~ N-th masking section 716, and so on. Between the two conjugate masking sections (i.e., first masking sections 710 and 714, or ~~N'th~~ N-th masking sections 712 and 716) is connected the adder 742 to 744 that add the two bits output from the masking sections in response to the input clock. Here, the output signals of the summers 742 to 744 have an I-channel component.

Please amend the paragraph starting on page 15, line 16 as follows:

Once an initial value for the primary scrambling code is applied to the first and second shift register memories 700 and 705 each having 18 registers for cyclically shifting the register value "a_i" or "b_i", the ~~0'th~~ 0-th register values of the first and second shift register memories 700 and 705 are fed into the adder 740 and the 18 register values "a_i" of the first shift register memory 700 are fed into the first to ~~N'th~~ N-th masking sections 710 to 712 in order to generate cyclically shifted sequences of the first shift registers. Meanwhile, the 18 register values "b_i" of the second shift register memory 705 are fed into the first to ~~N'th~~ N-th masking sections 714 to 716 in order to generate cyclically shifted sequences of the first shift registers. Then, the first masking section 710 masks the input values from the first (upper) shift register memory 700 (all 18 bits from 18 registers in the shift register memory 700) with a mask function $k^1{}_i$ (i.e., $\sum(k^1{}_i \times a_i)$) and outputs the masked values to the summer 744 for generating the first secondary scrambling code. The masking is concurrently processing in every masking sections 710 – 712. The ~~N'th~~ N-th masking section 712 masks the input values from the first (upper) shift registers with a mask function $k^N{}_i$ (i.e., $\sum(k^N{}_i \times a_i)$) and outputs the masked values to the summer 742 for generating the ~~N'th~~ N-th secondary scrambling code–. The ~~N'th~~ N-th masking section 716 masks the input values from the second (lower) shift registers with a mask function $s^N{}_i$ (i.e., $\sum(s^N{}_i \times a_i)$) and outputs the masked values to the summer 744 for generating the ~~N'th~~ N-th secondary scrambling code. The first masking section 714 masks the input values from the register memory 705 with a mask function $s^1{}_i$ (i.e., $\sum(s^1{}_i \times a_i)$) and outputs the resulting values to the adder 742 for generating the first secondary scrambling code. Each of the masking sections 710 – 712 masks the input values from the first shift register memory 700 and outputs the masked value to the respective adders 742 – 744. . Then, the adder 740 adds the output bits from the ~~0'th~~ 0-th registers of the first and second shift register memories 700 and 705. These generated output signals are immediately delayed at the delay 720. The adder 744 adds the output bits from the ~~N'th~~ N-th masking sections 712 and 716 to generate I-channel signals, which are immediately fed into the delay 724. The delay 722 delays the I-channel signals output from the adder 744 for a predetermined number of chips to generate Q-channel scrambling signals. The adder 742 adds the output bits from the first masking sections 710 and 714 to generate I-channel signals. These I-channel signals are immediately delayed for a predetermined number of chips at the delay 722. Then, the ~~0'th~~ 0-th and seventh register values of the first shift register memory 700 are added at the summer 730 and the added value is inputted to the seventeenth register, as the left-sided values are shifted to the right side by one and the utmost left-sided register is newly filled with the output value of the summer 730. The ~~0'th~~ 0-th, fifth, seventh, and tenth register values of the second shift register memory 705 are added at the adder 735, the added value is inputted into the seventeenth register, as the left-sided values are shifted to the right side by one and the utmost left-sided register (i.e., the seventeenth register) with the output value of the summer 735. This procedure is repeated to generate multiple scrambling codes.

Please amend the paragraph starting on page 17, line 10 as follows:

Referring to Fig. 8, once an initial value for the primary scrambling code is applied to a first shift register memory 840 having 18 upper shift registers and a second shift register memory 845 with 18 lower shifter register, the ~~0'th~~ 0-th register values of the first and second shift register memories 840 and 845 are fed into an adder 810. The output of the adder 810 is a primary scrambling code. The 18 register values "a_i" of the first shift register memory 840 are fed into a masking section 820. Meanwhile, the 18 register values "b_i" of the second shift register memory 845 are fed into a masking section 825. Then, the masking section 820 masks the input values from the first shift register with a mask function k_i (i.e., $\sum(k_i \times a_i)$) and outputs the masked values to an adder 815 for generating the first secondary scrambling code. The masking section 825 masks the input values from the second (lower) shift register with a mask function s_i (i.e., $\sum(s_i \times a_i)$) and outputs the masked values to an summer 815 for generating the secondary scrambling code. Then, the adder 810 adds the output bits from the ~~0'th~~ 0-th registers of the first and second shift register memories 800 and 805 to generate I-channel primary scrambling code signals. These I-channel primary scrambling code signals are immediately delayed for a predetermined number of chips at a delay 830 to generate Q-channel primary scrambling code signals. The adder 815 adds the output bits from the masking sections 820 and 825 to generate I-channel primary scrambling code signals, which are immediately delayed at a delay 835. Then, the ~~0'th~~ 0-th and seventh register values of the first shift registers are added at the adder 800, and the added value is output to the seventeenth register, as the left-sided values are shifted to the right side by one. The ~~0'th~~ 0-th, fifth, seventh and tenth register values of the second shift registers are added at the adder 805, and the added value is output to seventeenth register, as the left-sided values are shifted to the right side by one. This procedure is repeated to generate multiple scrambling codes.

Please amend the paragraph starting on page 18, line 25 as follows:

Referring to Fig. 9, when M secondary scrambling codes correspond to one primary scrambling code, the first, ~~(M+2)'th~~ (M+2)-th, ~~(2M+3)'th~~ (2M+3)-th, ..., ~~((K-1)*M+K)'th~~ ((K-1)*M+K)-th , ..., and ~~(511M+512)'th~~ (511M+512)-th gold codes are used as primary scrambling codes. The secondary scrambling codes corresponding to the ~~((K-1)*M+K)'th~~ ((K-1)*M+K)-th gold code used as the ~~(K)'th~~ (K)-th primary scrambling code are composed of M gold codes, i.e., ((K-1)*M+(K+1)), ((K-1)*M+(K+2))..., and (K*M+K)-th gold codes. Here, with 512 primary scrambling codes used, each of the secondary scrambling code sets corresponding to the 512 primary scrambling codes is composed of M secondary scrambling codes. In this structure, if a cell uses one of the primary scrambling codes then secondary scrambling codes belonging to the secondary scrambling code group corresponding to the primary scrambling code will be used_when the secondary ~~secondry~~ scrambling codes need to be used. As shown in Fig. 9, once a primary scrambling code is selected, the secondary scrambling codes corresponding to the primary scrambling code are generated by the adding cyclically shifted first m-sequences and the second m-sequence. Here, the secondary scrambling codes are generated through application of mask functions to the sequences in the first shift register memory. This method is adapted to a scrambling code generator of a transmitter as illustrated in Fig. 10, which concurrently generates one primary scrambling code and multiple secondary scrambling codes-.

Please amend the paragraph starting on page 20, line 24 as follows:

The adder 1030 adds the ~~0'th~~ 0-th register values of the first and second shift register memories 1040 and 1045 to generate a scrambling code, which becomes a primary scrambling code. The adders 1032 to 1034 each adds one bit generated from the masking sections 1000 to 1005 to one bit generated from the second shift register memory 1045, respectively, to generate I-channel scrambling code signals. Here, the output from the adder 1030 is used as the primary scrambling code and the scrambling codes output from the adders 1032 to 1034 can be used as secondary scrambling codes that corresponds to the primary scrambling code. The following is an example of possible mask values ($k^1{}_i$ to $k^n{}_i$): $k^1{}_i$ =(000000000000000010), $k^2{}_i$ = (000000000000000100), $k^3{}_i$ =(000000000000001000)..... By controlling the mask values, other primary and secondary codes can be generated. The following example shows how to obtain a necessary mask code to cyclically shift a m-sequence -n- times. In general, divide x^n by the generation polynomial for the m-sequence (i.e., $x^n/f(x)$) and take the remainder of the division to form the mask code. For example, if a mask code that cyclically shifts 31 times is desired, take x^{31} and divide it by $f(x) = x^{18} + x^7 + 1$ the generation polynomial and find the remainder which cannot be divided further. The final remainder is $x^{13} + x^9 + x^2$ as shown by the following:

Please amend the paragraph starting on page 22, line 3 as follows:

Once an initial value for the primary scrambling code is applied to the first and second shift register memories 1040 and 1045 each having 18 registers, the ~~0'th~~ 0-th register values of the first and second shift register memories 1040 and 1045 are fed into the adder 1030 and the 18 register values "a_i" of the first shift register memory 1040 are fed into the first to ~~N'th~~ N-th masking sections 1000 to 1005 in order to generate 1 to N cyclically shifted sequences of the first m-sequence. Then, the first masking section 1000 masks the input value(a_i) from the first (upper) shift register memory 1040 with a mask function $k^1{}_i$ for generating the first secondary scrambling codes (i.e., $\sum(k^1{}_i \times a_i)$) and outputs the masked value(a_i) to the adder 1032. The ~~N'th~~ N-th masking section 1005 masks the input value(a_i) from the first (upper) shift register memory 1040 with a mask function $k^N{}_i$ for generating the ~~N'th~~ N-th secondary scrambling codes (i.e., $\sum(k^N{}_i \times a_i)$) and outputs the masked values to the adder 1034. At the same time, the adder 1030 sums the output bits from the ~~0'th~~ 0-th registers of the first and second shift register memories 1040 and 1045. The generated output signals are immediately delayed at the delay ~~1022~~ 1020. The adder 1032 sums the output bits from the first masking section 1000 and the ~~0'th~~ 0-th shift register of the second shift register memory 1045. The output signals are immediately fed into the delay 1022. Thereafter, the ~~0'th~~ 0-th and seventh register values of the shift register memory 1040 are added at the adder 1010 and the adder 1010 outputs the sum to the seventeenth register, as the left-sided values are shifted to the right side by one and the utmost left-sided register is newly filled with the output value of the adder 1010. The 0-th, fifth, seventh and tenth register values of the shift register memory 1045 are added at the adder 1015, and the adder inputs the sum into the seventeenth register of the register memory 1045 as the left-sided values are shifted to the right side by one to fill the utmost left-sided register (i.e., the seventeenth register) with the output value of the adder 1015. This procedure is repeated to generate multiple scrambling codes.

Please amend the paragraph starting on page 23, line 11 as follows:

Referring to Fig. 11, once an initial value for the primary scrambling code is applied to a first shift register memory 1140 having 18 registers and a second shift register memory 1145 with 18 registers~~.~~, the ~~0'th~~ 0-th register values of the first and second shift register memories 1140 and 1145 are fed into an adder 1120. The 18 register values "a_i" of the first shift register memory 1140 are fed into the masking section 1100 in order to generate a cyclically shifted m-sequence. Then, the masking section 1100 masks the input values(a_i) from the register memory 1140 with a mask values k_i for generating the first secondary scrambling codes (i.e., $\sum(k_i \times a_i)$) and outputs the masked values to an adder 1125. The adder 1120 sums the output bits from the ~~0'th~~ 0-th registers of the first and second shift register memories 1140 and 1145. The output signals of the adder 1120 are immediately delayed at a delay 1130. Meanwhile, the adder 1125 sums the output bits from the masking section 1100 and the ~~0'th~~ 0-th shift register of the second shift register memory 1145 and outputs the sum to a delay 1135 immediately. Then, the ~~0'th~~ 0-th and seventh register values of the first shift register memory 1140 are added at the adder 1110, in which case the left-sided values are shifted to the right side by one and the utmost left-sided register is newly filled with the output value of the summer 1110. The ~~0'th~~ 0-th, fifth, seventh and tenth register values of the second shift register memory 1145 are added at the adder 1115, shifting the left-sided values to the right side by one and newly filling the utmost left-sided register with the output value of the adder 1115. The mask values can be controlled by a controller_(not shown) when the receiver needs to generate other scrambling codes.